AD-A040 195 NORTH CAROLINA STATE UNIV RALEIGH SIGNAL PROCESSING LAB F/G 9/2
RUN LENGTH CODING FOR IMAGE ANALYSIS.(U)
JAN 77 J N ENGLAND DAAG29-76-G-0133
UNCLASSIFIED SPL-17 ARO-13989.2-EL NL

| OF |
AD
A040 195

END

DATE
FILMED
6-77

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ARO-13989.2-EL

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
RUN LENGTH CODING FOR IMAGE ANALYSIS

5. TYPE OF REPORT & PERIOD COVERED
TECHNICAL ~~MEMO~~ memo,

6. PERFORMING ORG. REPORT NUMBER
SPL-17

7. AUTHOR(s)
J.N./England

8. CONTRACT OR GRANT NUMBER(s)
DAAG 29-76-G-0133

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Electrical Engineering Dept.
NC State University
Raleigh, NC 27607

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
U. S. Army Research Office
Post Office Box 12211
Research Triangle Park, NC 27709

12. REPORT DATE
~~1/77~~ Jan 77

13. NUMBER OF PAGES
~~17~~ 19

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
NA

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

ARO
13989.2-EL

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SCENE ANALYSIS, RUN LENGTH CODING, IMAGE ANALYSIS, COMPUTER VISION

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Image coding techniques which have proven to be useful for transmission purposes have seldom been applied to image analysis. This report examines the application of one such technique, run-length encoding, to problems in the scene analysis domain. Several operations within a run-length encoded data structure are discussed, including area access, region growing, edge following, and corner identification.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 831

SIGNAL PROCESSING LABORATORY

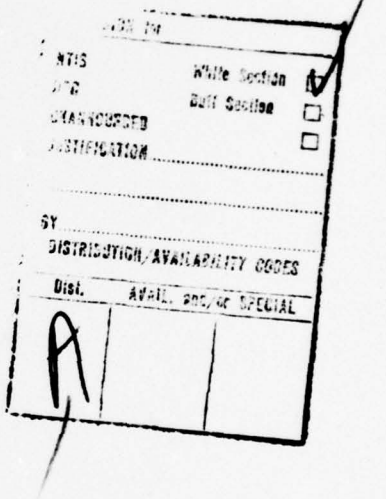
REPORT #17

Run Length Coding for Image Analysis

by

J.N. England

January 1977



NORTH CAROLINA STATE UNIVERSITY
ELECTRICAL ENGINEERING DEPARTMENT

This work was supported by Grant DAAG 29-76-G-0133

U.S. Army Research Office

ABSTRACT

Image coding techniques which have proven to be useful for transmission purposes have seldom been applied to image analysis. This report examines the application of one such technique, run-length encoding, to problems in the scene analysis domain. Several operations within a run-length encoded data structure are discussed, including area access, region growing, edge following, and corner identification.

I. Run Length Coding of Images

A. Introduction

The basic data structure used to represent M x N images is a one-dimensional list of intensity values of length MN. This structure is necessary because almost all computers are incapable of any sort of an explicit operation on the most natural representation of an image, an M x N array of intensities. There are a few exceptions, of course, including the CLIP series of parallel array processors [1,2].

For a regular array (uniform length of rows and columns) indexing within the list is fairly easy. For instance, a doubly subscripted Fortran variable, I(X,Y) is the ((Y-1)*N)) +X-1 member of the list I representing an M x N array ($1 \leq X \leq M$), ($1 \leq Y \leq N$).

For those arrays which do not contain uniformly dimensioned rows or columns (and, including those which do) a base-displacement method of addressing is required. The starting address of row Y is used as a base and X is used as a displacement in order to find the address of element

I(X,Y)=I ((address Y)+X). This is a much faster method of indexing assuming an indirect fetch is faster than a multiply and two decrements. This method does require some sort of list of the addresses of all rows, yielding for a regular array N(M+1) rather than NM storage requirements.

That compression of an image for computer analysis differs from compression of the image for transmission can be seen in the discussion above. Even though storage size is increased by base-displacement addressing, ease of computation on elements of the image is increased. Compaction schemes for transmission purposes are oriented toward minimum storage size (in bits, not words) and minimum mean square error image degradation. These goals may yield methods quite unsuited for image analysis work. It is very difficult to entertain the notion of region growing, for example, in a differential entropy-coded image. Some of the methods of image compression for transmission such as Hadamard or Fourier transforms are useful in a feature extraction or enhancement context but the overhead involved in converting to the transform domain and back may be considerable.

Image compression by run-length coding is an idea that has been around for some time [3,4,5]. Its use, however, has been largely restricted to facsimile transmission [6,7], although recently several computer graphics systems have employed run-length coding to reduce storage requirements [8,9,10].

B. Methods

The most natural form of run-length coding for raster-type images is horizontal scan line encoding. The image array is scanned (typically, left-to-right) and an intensity code and run-length are stored whenever the current pixel intensity differs from the previously stored intensity code by some measure. If, for example, the measure is simply the absolute difference between current and previous intensities and a decision is made to store whenever this measure is greater than zero an exact copy of the image is stored. If two words (intensity and run-length) must be stored at a time, no storage savings can result unless the average of the run-lengths ($\overline{RL}$) is greater than two. Thus the amount of storage savings is highly image and noise dependent. If, however, a length code is stored only if it is greater than one (and identified by a flag bit) the worst case performance of the compaction scheme is exactly the same as the unencoded image! Table 1 lists $\overline{RL}$ and storage savings obtained from several 256 x 256 8-bit intensity image for different decision thresholds. Several theoretical predictions of bit storage savings (11,12) for run-length coding have been made previously. In some applications environments the savings may be significantly greater than those indicated above. For example, in an industrial environment a uniform background can be provided behind the object in view of the television camera(s). This low information background can be encoded into a relatively small storage area.

Table 1 256 x 256 = 65536 pixels

Threshold	# Words	Savings
Ø	64056	1.02:1
4	37326	1.76:1
8	30539	2.15:1
16	15598	4.20:1
32	8008	8.18:1

It should be noted that several schemes can be used in the storage decision process. Besides difference between stored and current pixel values, the difference between current and previous pixel values $|I(X,Y)-I(X-1,Y)|$ can be used. In fact, the decision process is precisely that of edge finding and Roberts [13] Laplacian, Hueckel [14], recursive filtering [15]., etc. methods may be applied.

The best methods from a hardware implementation standpoint are those involving a local two-dimensional computation or those which can be applied on an image in raster scan order. The choice of a two-dimensional measure such as a modified gradient yields the two components:

$$D_Y = |I'(X,Y-1) - I(X,Y)|$$

$$D_X = |I'(X-1,Y) - I(X,Y)|$$

where $I'(i,j)$ is a previously encoded intensity and $I(i,j)$ is the current pixel intensity. In order to achieve some coherency in the vertical direction the procedure in Fig. 1 is used. (THX and THY are threshold values). Essentially, whenever a new run is started, its intensity value is that of the region above if D_Y is small, and that of the pixel itself if D_Y is large. This procedure is a mild bottom-up

region grower and greatly simplifies later edge location and region growing processes. Again, from an applications standpoint this method is well suited to the types of objects potentially found in an industrial scene.

As discussed earlier, data compaction for image analysis is not a simple matter of number of words or bits saved. From an access ease standpoint the run length coded image offers a disadvantage for access to random elements. A list of pointers to the beginnings of image rows is kept separately for indirect addressing of Y. To find the value at a particular X location, repeated additions of run lengths must be made and compared to the desired X location. If a variant of the coding scheme is used and final X location in a run is coded rather than run length, the horizontal access problem is reduced to a search. The search can be speeded considerably by noting that line-to-line coherency of the image implies that a corresponding X location in an adjacent line can be found easily. If X is within the i^{th} segment (run) of line Y, X will be in or close to the i^{th} segment of lines Y+1 and Y-1. Thus, operations within a region located about some X,Y can proceed with little difficulty in access. The run-length scheme offers quite an advantage to computation involving horizontal elements since pixel redundancy is indicated by the run length. A major advantage of the run length coded data structure over other methods of reducing storage area lies in the maintenance of positional accuracy. That is, we can achieve a reduction in storage size by run length coding but maintain fill horizontal and vertical resolution with only

a minimal loss of intensity data and with a first order edge extraction and region growing procedure in the bargain!

II. Operations on Run Length Coded Images

A. Access

We shall consider a data structure representing an image in run length coded form as follows:

ILIST - A list of intensity values of each run length segment

RLIST - A list of the X coordinate of the last pixel in each run length segment.

YLIST - A list of pointers to the first entry in RLIST for each horizontal line.

In order to find the intensity value at a given pixel location X,Y we create a pointer PTR=YLIST(Y). If RLIST(PTR)$\geq$X then ILIST(PTR) contains the desired intensity value. Otherwise we increment PTR and check again, thus linearly searching a section of RLIST. In order to speed the search we could adopt a binary search technique to some desired level.

As mentioned previously this is not an efficient data structure for random access. However, once we know PTR for some X,Y we can easily find the intensities for horizontally nearby pixels by incrementing or decrementing PTR as necessary. Likewise vertically nearby pixel intensities can be found by creating NEWPTR=YPTR(NEWY)+OLDPTR-YPTR(OLDY) and incrementing or decrementing as necessary. The coherency of the picture should ensure that this search will be a short one. While this seems like a cumbersome procedure it should be pointed

out that every ILIST entry contains the value of not just a single pixel but possibly of many pixels and that the search process gathers data about an entire region of the image and not just the few selected pixels.

B. Region Growing

A form of region growing may be performed on the run length coded image. As shown before, the original coding scheme is a mild form of bottom-up region growing. We may extend this procedure into the coded data structure. If, for some segment on line Y which ends at location X, we examine the intensity value of a corresponding segment on line Y+1 and find little difference between the two we may reasonably merge them into a single region if and only if there is some overlap between the two segments (see Fig. 2). This procedure may be carried out through the image creating tags for run length segments assigning them to regions.

A clean-up operation should then be performed to merge isolated segments which have no good assignment to adjacent regions. We may then assign them to the least objectionable adjacent region as they are very probably artifacts of the low level coding merge and out previous merging operation.

C. Boundary Mapping

Creating a boundary map from a run-length coded image data structure is fairly straightforward. RLIST (original as modified by region growing) consists of all the horizontal boundary points. We may find the vertical boundary points by starting at line Y=2. If we examine the difference between a segment on line Y+1 and line Y and find that it exceeds some

selected threshold (the same operation as in bottom-up region growing) at any point(s) along the segment we then add this point(s) to the list of vertical boundary points (if separate from the horizontal boundary point list).

If the vertical boundary point operation is carried out in an orderly top-to-bottom, left-to-right manner, the horizontal boundary points are obtained concurrently (since we are accessing segments for comparison) and the end result is an ordered list of boundary points which may then be used as desired.

D. Texture

The mild region growing and edge extraction performed during the encoding process suggests a method of measuring texture within some selected area of an image.

Throughout this selected area we perform the bottom-up region growing discussed in (B) above with a threshold equal to zero, omitting the isolated segment merge of (B), and then average the number of distinct regions found over the selected area. Finally, we use this average as a measure of texture. Of course this measure depends upon the thresholds used in the run length encoding process, but otherwise does not depend upon the actual region values.

The region identification process above is necessary because while the encoding process does take into account vertically adjacent pixels, there is unfortunately no way to indicate this explicitly in the encoded data structure without developing a completely separate vertical run length encoded version of the original image.

E. Edge Following

Following an edge through an image is often an important step in image analysis. A procedure which follows a straight edge within our run length encoded image data structure will be described.

During the initial encoding process, true edges can very often be represented as several closely spaced edges as in Figure 3. The first step in edge following will be to merge the short run lengths which are created by these closely spaced edges. A procedure for this merging operation is outlined in Figure 4. A similar procedure must be carried out in the vertical direction. Unfortunately, this is not so simple as in the horizontal direction but is still straightforward.

Once these extraneous edges are reduced by the preceeding process we proceed to follow the edge by using a slope computation process. To form our first estimate of edge slope we take the edge location X(Y) for row Y and the edge location X(Y+1) for line Y+1 and compute S(1=X(Y)-X(Y+1). We compute S(2) in like manner as (X(Y)-X(Y+2))/2. We use the average of S(1) and S(2) as our first estimate of the actual edge slope. We use this to preduct X(Y+2) in order to speed our access process since the intensity of the predicted X'(Y+2) identifies our prediction as being to the left or right of the actual edge. We find the actual X(Y+2) and use this to computer S(3) and a new estimate of the slope and proceed to line Y+3. We follow this procedure until termination

of the edge. Termination is detected by any of several conditions:

(1) the image boundaries are exceeded,

(2) S(J) and S(J+1) both differ from the estimated slope by some threshold indicating the edge has changed direction,

(3) the predicted X'(J) lies in a run length with an intensity which differs from that of both regions lying on either side of the edge being followed indicating an edge has terminated.

F. Feature Extraction

In order to find small regions of the run length coded which are likely to contain features of interest we follow the ideas of Pingle and Thomas [16]. As a first step we should apply the multiple edge eliminator described in the previous section on edge following. Then we identify areas with a high variance for further consideration. Areas so identified are subjected to a corner detector operation.

Unlike most image representations, the run length coded data structure does not lend itself most easily to square or rectangular arrays. We will therefore consider a small area of a fixed number of lines N but not necessarily a fixed number of pixels per line. We shall decide that our rows will include at least M pixels and will include complete run lengths, as illustrated in Figure 5. Our areas will overlap each other by N/4 lines and by one run length per line. They will be arranged as in Figure 6 so that at least an MXN array will be covered by each area.

Calculation of the variance of an area proceeds quite straightforwardly. We first calculate the mean intensity $\bar{I} = (\Sigma R(J)I(J))/\Sigma R(J)$ where R(J) is the run length of segment J and I(J) is the intensity of segment J.

The variance is $(\Sigma R(J)(I(J)-\bar{I}))/\Sigma R(J)$. If the variance exceeds some given threshold we decide this area is interesting enough to warrant further investigation, otherwise we move onto the next area.

For interesting areas we apply a corner detector. A corner is defined by the abrupt change in direction of an edge or by the intersection of two or more edges. We shall modify the edge following technique described earlier because of the small size of the area involved and because of the likelihood that a corner is within the region. The measure of slope we shall use is $X(Y)-X(Y+1)+(X(Y)-X(Y+2))/2$. This is the initial estimate of slope referred to earlier. However, in the present context we shall not make further estimates but shall proceed to line Y+2 and recalculate the slope. If the change in slope exceeds some threshold we surmise that a corner or intersection is present. This procedure must be carried out, of course, for the edges of each region within the area under consideration.

G. Cross-correlation of Feature Areas

Once we have found two feature areas using the algorithm of the last section, we can compute their cross-correlation. Since the two areas have the same number of pixels per line we must describe the area over which the cross-correlation will be taken as in Figure 7. The X_{min}

and X_{max} for a given line are endpoints of a segment in one or the other area and we are thus insured that we will take the cross-correlation over at least an MxN area.

The cross-correlation is defined as $\sum_I A(I)B(I)/\Sigma(I)$ where A(I) and B(I) are the pixel intensities of areas A and B respectively. Computation of this measure may be speeded by taking advantage of the run length coding to compute the product over segments rather than over individual pixels. Figure 8 illustrates a manner of doing so. The determination of segment lengths is, in fact, not as complicated as might be supposed since each endpoint location is already explicitly stored in the corresponding RLIST for each area.

III. Implementation

The preceeding methods developed for operations in run length coded images will be implemented in the Signal Processing Laboratory during Spring 1977. We expect this to point out problems in the methods and to lead to improved versions of them. Basically, the ideas developed above are the beginning nucleus for the development of a system for scene analysis in encoded images. Future publications will deal with the results obtained by implementation and the modified operations which result during the learning process.

REFERENCES

[1] M.J.B. Duff, "Clip 4: a large scale integrated circuit array parallel processor", Proc. 3rd Int'l. Joint Conf. on Pattern Recognition, Coronado, CA, IEEE 76CH1140-3C (Nov. 1976) 728-733.

[2] M.J.B. Duff, et.al., "A cellular logic array for image processing", Pattern Recognition (1973) 229.

[3] G.G. Gouriet, "Bandwidth compression of a television signal", Proc. IRE (May 1957) 265-272.

[4] J.O. Limb and I.G. Sutherland, "Run-length coding of television signals", Proc. IEEE (Feb. 1965) 169-170.

[5] A.H. Robinson and C. Cherry, "Results of a prototype television bandwidth compression scheme", Proc. IEEE (March 1967) 356-364.

[6] T.S. Huang, "Run length coding and its extensions", in Picture Bandwidth Compression (1972) 231-264.

[7] D.C. vanVoorhis, "An extended run-length encoder and decoder for compression of black/white images", IEEE Trans. on Inf. Theory, IT-22 (March 1976) 190-199.

[8] A.J. Myers, "A digital video information storage and retrieval system", Computer Graphics (Summer 1976) 45-50.

[9] B.A. Laws, "A gray-scale graphic processor using run length coding", Proc. Conf. on Comp. Graphics, Pattern Recognition and Data Structure (May 1975) 7-10.

[10] J.F. Eastman, "An efficient scan conversion and hidden surface removal algorithm", Computers & Graphics (1975).

[11] W.K. Pratt, "Coding compression of a television bandwidth reduction system", Proc. IEEE (June 1966) 914-916.

[12] K.G. Gray and R.S. Simpson, "Upper bound on compression ratio for run-length coding", Proc. IEEE (Jan 1972) 148.

[13] L.G. Roberts, "Machine perception of three-dimensional solids" in Optical and Electro-Optical Processing of Information (1965) 159-197.

[14] M. Hueckel, "A local visual operator which recognizes edges and lines", Journal ACM (Oct. 1973) 634-647.

[15] J.W. Modestino and R.W. Fries, "Edge detection in noisy images using recursive digital filtering", 1976 Joint Workshop on Pattern Recognition and Artificial Intelligence (June 1976) IEEE 76CH1169-2C. 84-89.

[16] K.K. Pingle and A.J. Thomas, "A fast, feature-driven stereo depth program" Stanford AI Lab AIM-248 (May 1975).

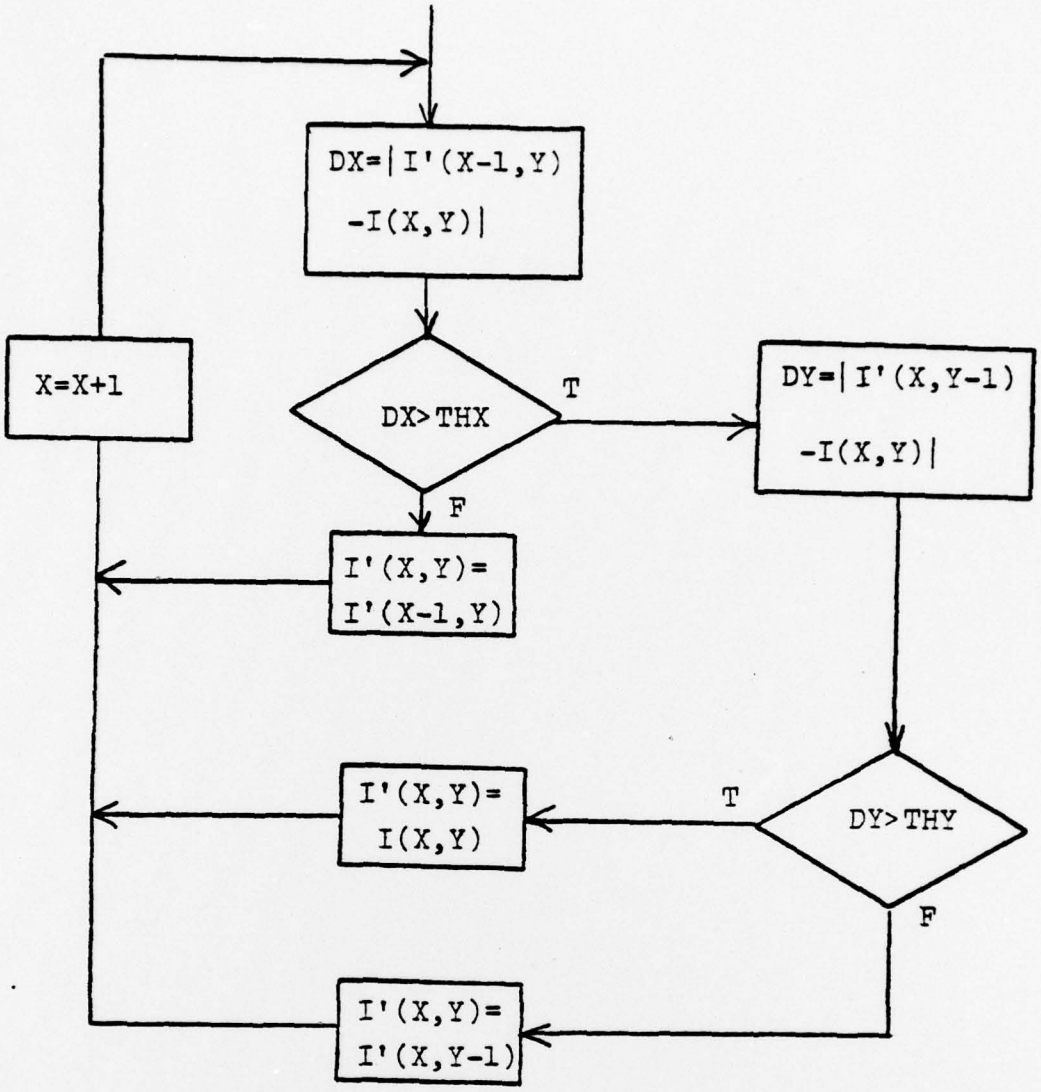


Figure 1 Run length coding procedure

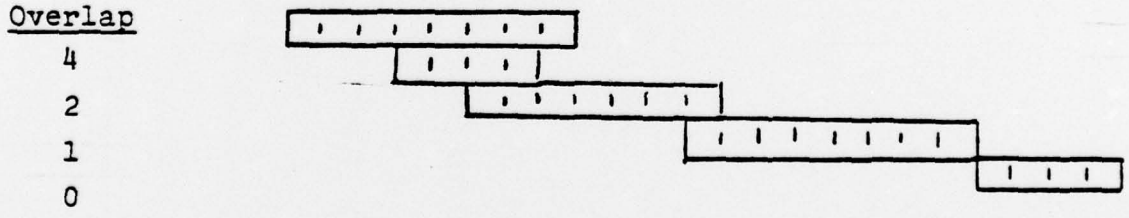
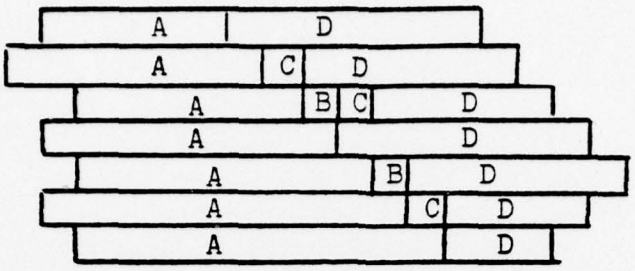


Figure 2 Overlap of region segments



A,B,C,D intensities A<B<C<D

Figure 3 Extraneous edges between regions

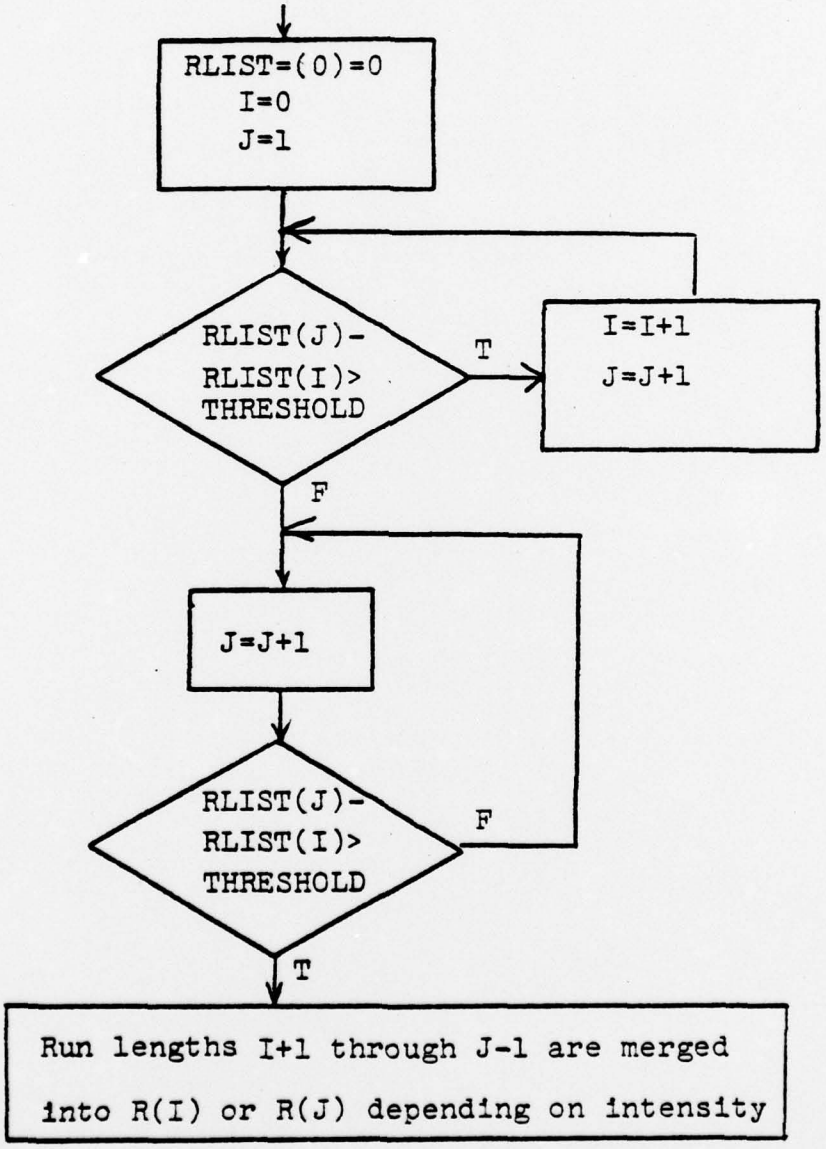


Figure 4 Extraneous edge removal

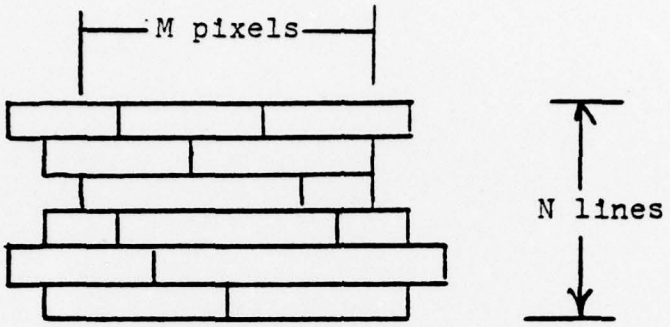


Figure 5 Feature area

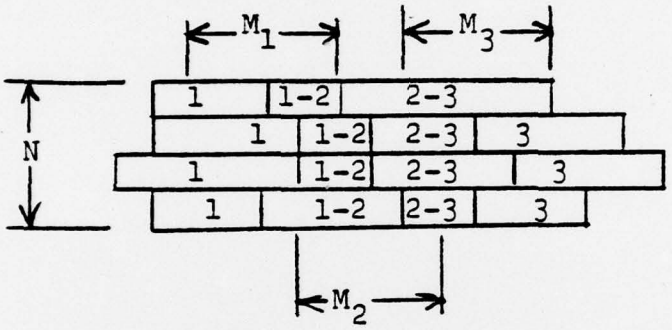


Figure 6 Feature area overlap

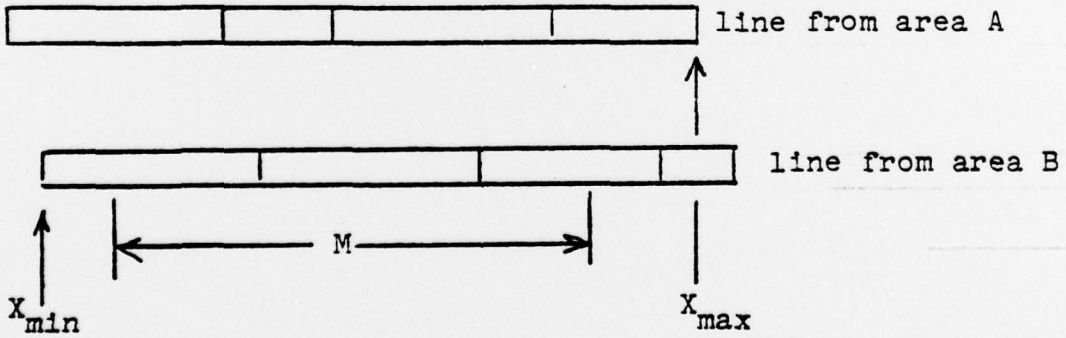
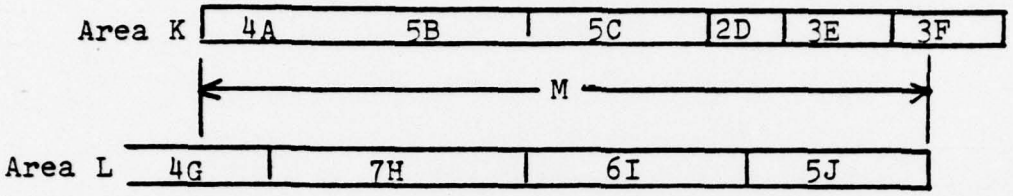


Figure 7 Correlation length selection

where 5C represents an intensity of C
for a run length of 5



$$\Sigma K(X)L(X) = 2AG + 2AH + 5BH + 5CI + 1DI + 1DJ + 3EJ + 1FJ$$

as opposed to

$$\Sigma K(X)L(X) = AG+AG+AH+AH+BH+BH+BH+BH+BH+CI+CI+CI+CI+CI+$$
$$DI+DJ+EJ+EJ+EJ+FJ$$

Figure 8 Correlation computation